1 – three.js Introduction

Pedro Tavares – 93103 Ricardo Cruz – 93118

MSc Robotics and Intelligent Systems MSc Communication and Web Technologies

Information Visualization, University of Aveiro, 2022

First Example

The first exercise from the practical class focuses on the essential aspects of a three.js program. How to create a scene, a camera, and an object and some simple parameters that can describe these elements. For this to happen, all the elements must be created and then, both the camera and the objects must be added to the scene.

Finally, the scene gets rendered by the WebGLRenderer object from the Three.js library. This object is used in the *render* function that besides creating the initial renderization, is responsible for animating the scene, which in this case is a cube in constant rotation around the x and y axes.

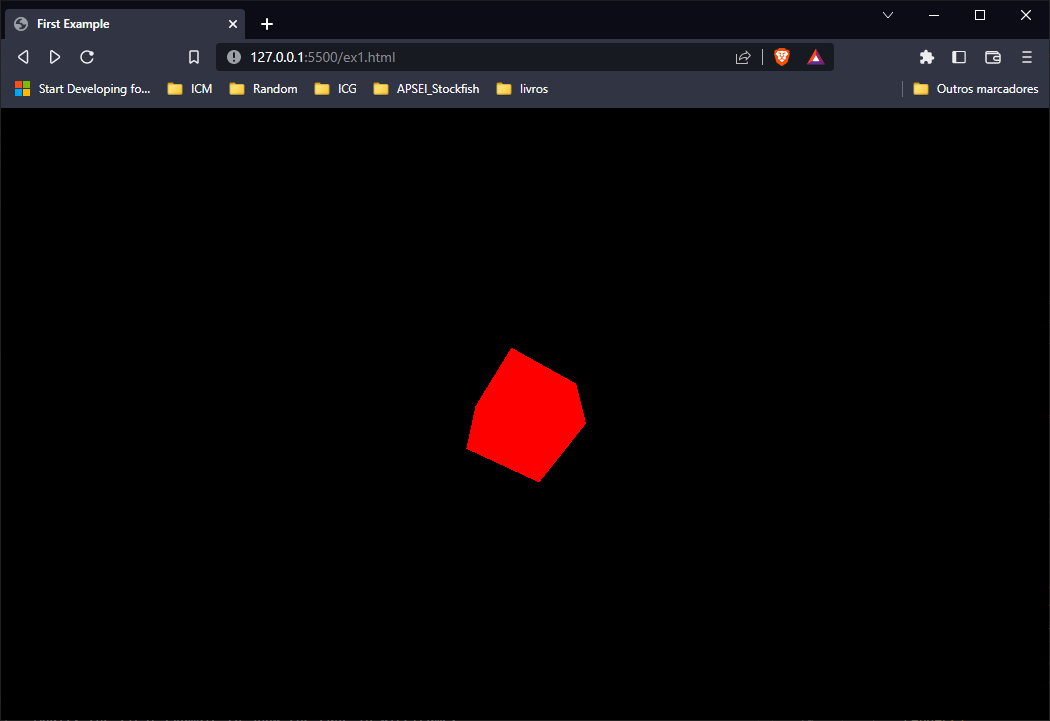


Figure : Output from ex. 1.

# 2D Primitives

For the second exercise, the objective is to manipulate 2D geometries with the Three.js library. For this, it is used the *BufferGeometry* class that, in this case, is used to define a triangle through the points of the vertices. Besides this, the background of the scene used has a flat color which was implemented through the *setClearColor* function with the desired color as its argument.

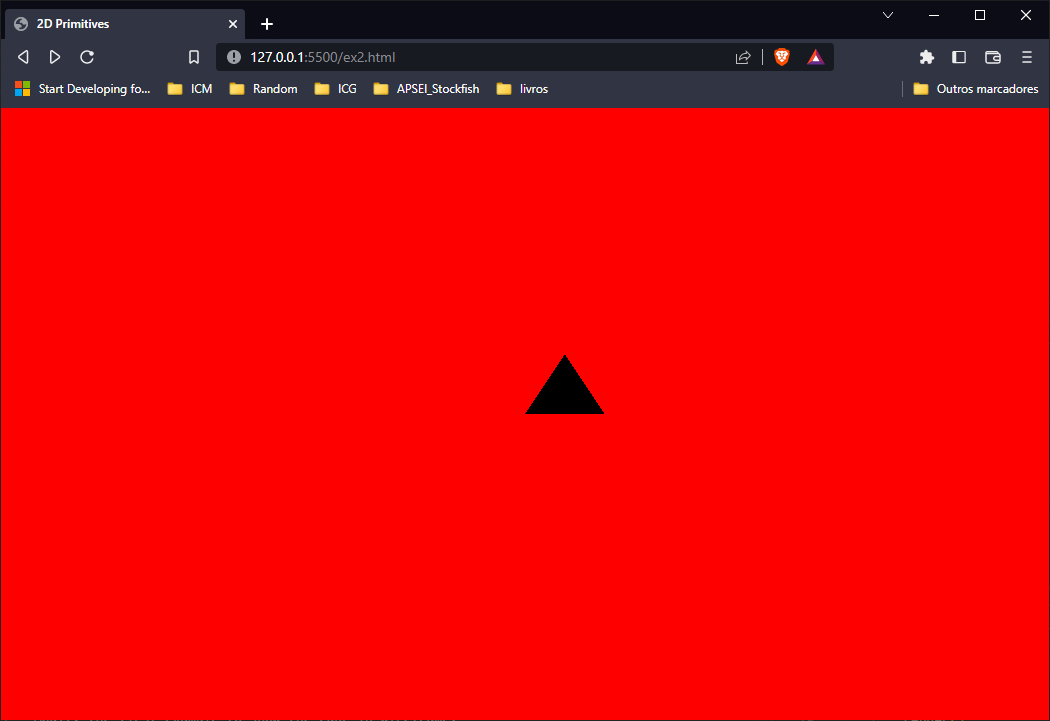


Figure : Output from ex. 2.

# Addition of color

In this exercise, four triangles were created with the same *BufferGeometry*. Four different triangles were created with different attributes, from position to color, and since some triangles were not visible in the scene because they were not “facing” the camera, another attribute was added, which is the DoubleSide attribute, which means that, no matter the side that the object is facing, its sides should be visible. Besides the color change and the visibility, a wireframe triangle was introduced. The wireframe attribute means that only the edges of a certain object will be displayed on the scene.

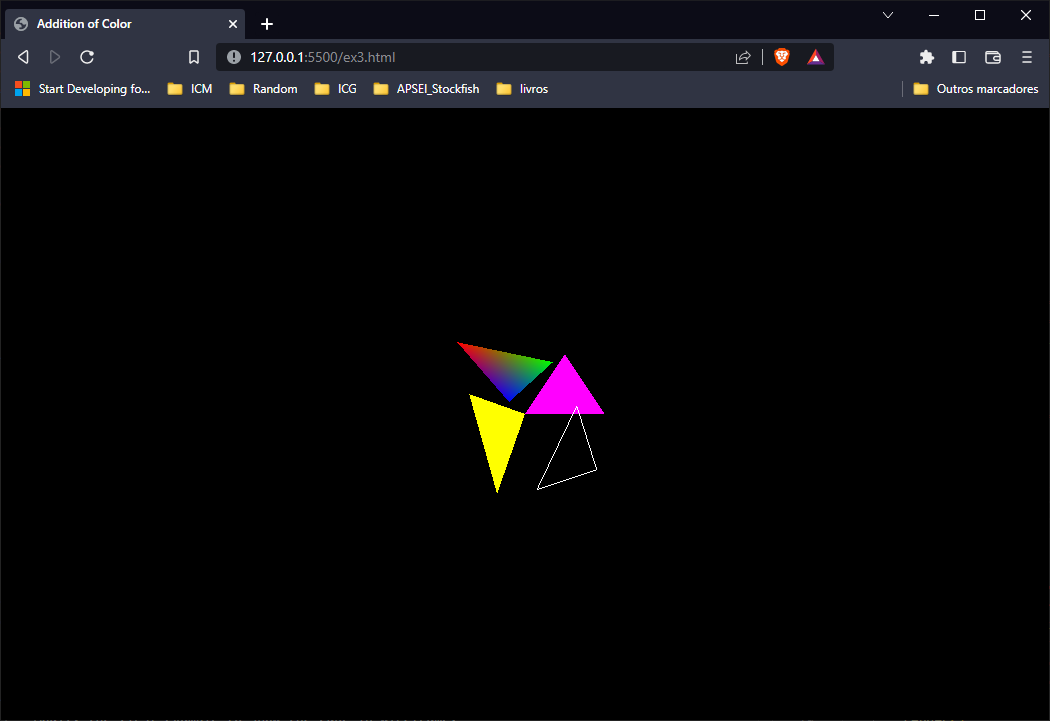


Figure : Results from ex. 3.

# Viewport Update

This exercise is built on top of the first one and solves an issue with window resizing. Before this exercise, if the window were to change size, the visualization would not adapt, and this exercise represents the answer to this one problem. To solve it, an event listener is added to the resize event and inside that listener the renderer size is updated to match the current window size.

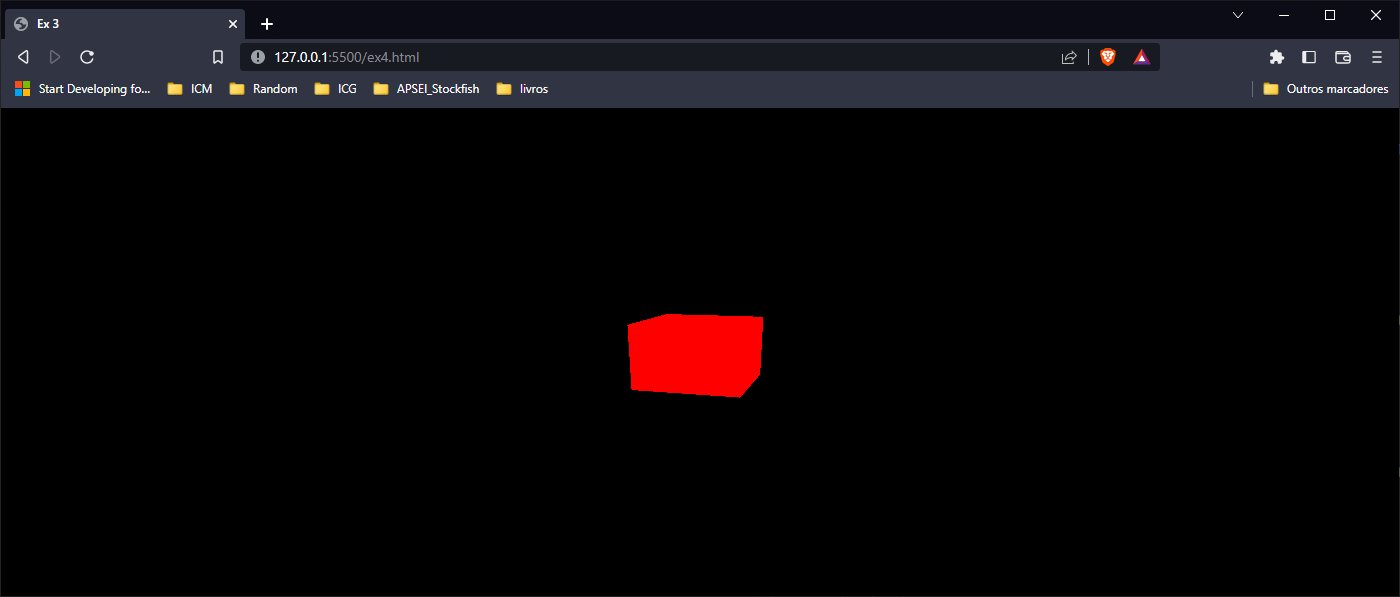


Figure 4: Results from ex. 4 after window resize.

# Other Primitives

For this section, again on top of the first example, we had to explore other geometries and properties of the objects. Initially, the cube from the first scenario was set to be represented as a wireframe, so the wireframe attribute was set to true. This led to the cube being shown as a red wireframe. Two other wireframe geometries were added. The octahedron and the capsule, both with no vertex colors which translates into white wireframes. The last objects to be added are a plane object, which was added with a depth of -3 so that it would be shown after the other geometries and serve as a background for the wireframe objects and a ring. The ring added to the scene was not a full ring, it was only a half and it was added with a yellow color between the plane object and the wireframe cube, so that the cube could be fully visible as well as the ring.

Both the capsule and the octahedron objects were added with custom constructor attributes and finally, the capsule, just like the cube is constantly rotating just like the cube from the first exercise.

Uma imagem com texto, captura de ecrã, eletrónica, apresentação

Descrição gerada automaticamente

Figure 5: Results from ex. 5.

1. References
2. Three.js. (2022). *CapsuleGeometry*. Obtido de <https://threejs.org/docs/api/en/geometries/CapsuleGeometry.html>
3. Three.js. (2022). *Octahedron Geomtry*. Obtido de [https://threejs.org/docs/api/en/geometries/OctahedronGeometry.html](https://threejs.org/docs/api/en/geometries/OctahedronGeometry.html%20)
4. Three.js. (2022). *PlaneGeometry*. Obtido de <https://threejs.org/docs/api/en/geometries/PlaneGeometry.html>
5. Three.js. (2022). *RingGeometry*. Obtido de <https://threejs.org/docs/api/en/geometries/RingGeometry.html>